Presented at ATM99 DEVELOPMENTS Conf, Mar 3 - Apr 1.1999, Rennes, France

Performance Measurement of Remote ATM Cluster[1]

Charlotte Martin[2], Alan Mink, Wayne Salamon, Mike Indovina and Michel Courson

Information Technology Laboratory
National Institute of Standards and
Technology
Gaithersburg, MD 20899
amink@nist.gov
(301) 975-5681 (office)
(301) 869-7429 (fax)

Resume

Le but de cette etude ést d'évaluer la mise en place et les performances de l'association de réseaux de PCs (clusters) distants, en combinant dynamiquement plusieurs clusters pour former un seul super-cluster à travers les réseaux locaux existants, voire même Internet. La mise en place d'une telle configuration est décrite, ainsi que la structure des réseaux empruntés, les machines utilizées se trouvant sur des sous-réseaux Ethernet et des switches ATM différents. On montre que les réseaux exclusivement commutés n'apportent qu'une très faible dégradation de performance, alors que les routeurs peuvent avoir une influence appréciable.

Abstract

We investigate the configuration and performance of remote commodity computing clusters. This is the dynamic pooling of separate clusters into a single large remote cluster via existing LANs or even the Internet. We discuss the configuration and setup of these remote clusters, as well as the networks since these clusters are on separate Ethernet subnets and separate ATM switches. We show that pure switching networks add little additional overhead to remote cluster computing applications, whereas routers can a have significant impact.

[1] Certain commercial items may be identified but that does not imply recommendation or endorsement by NIST, nor does it imply that those items are necessarily the best available for the purpose.

[2] Charlotte Martin was a university guest researcher at NIST from the Institut National des Telecommunications, Paris, France.

Introduction

As the number of commodity-based computing clusters [BEC95] increases, the interest to dynamically interconnect them and to form temporarily larger clusters also grows. By commodity clusters, we mean low-cost PCs or workstations interconnected by common switches or networks, usually running Linux or FreeBSD for the operating system. Such cluster computing has become a popular implementation of a distributed memory, message passing environment. A primary factor is the significant performance and low initial capital cost of these clusters. However, there are substantial support costs which tend to be overlooked initially. It has been estimated that one system administrator is required for each 16 to 64 nodes. Also, as the number of nodes exceeds 32, the number of hardware component failures becomes significant in administration of the system. Other costs are hidden in lost productivity and frustration for scientists and programmers who must track down "work-arounds" for non-integrated and unsupported software. These points and related aspects make smaller clusters of 16 to 32 nodes very popular.

As programs are successfully developed in a cluster environment, there is a strong tendency to pursue larger problem sizes. These then require more nodes and associated memory. Eventually, some researchers need more nodes than are available in their clusters. Many clusters may exist within an organization or at a number of collaborating organizations. It is reasonable to expect cooperation and temporarily pooling of the separate clusters into a single large remote cluster via existing LANs or even the Internet. This is the basis of our motivation to investigate remote clusters. This is more modest than "metacomputing" approaches to distributed computing that wish to use very large numbers of nodes via the Internet [GRI97, FOX97]. Where both approaches share some common problems, such as different administrative domains and software versions, "metacomputing" faces additional challenges, such as security and availability.

The NIST organization has a number of independent experimental clusters, and thus is a strong candidate for using remote clusters. We have evaluated the performance of commodity clusters [IND98] compared to some commercially available, integrated clusters, and also have evaluated both ATM/OC3 and Fast Ethernet technologies for these clusters [IND98]. An evolving path of this research leads us to investigate the performance of remote clusters connected via a pure ATM network, as well as our campus wide LAN, which consists of Ethernet segments interconnected via an FDDI backbone. We discuss the configuration and setup of these remote clusters as well as the networks since these clusters are on separate Ethernet subnets and separate ATM switches. Special attention is given to setting up Permanent Virtual Circuits (PVC) and Switched Virtual Circuits (SVC) channels on the ATM switches. To measure the performance of these various interconnected clusters we will use the NIST MultiKron® [MIN94,MIN97,MIN98] performance measurement instrumentation which is time synchronized [LEV95,MIL92], both locally and remotely, with NIST developed instrumentation based on the Global Positioning System (GPS). With this instrumentation we will be able to accurately measure the throughput and latency between these linked clusters for each of the various networks. Furthermore, we will run well-known benchmarks as well as some NIST application programs to determine their performance in these remote cluster environments.

1 Cluster configuration

To study the performance of ATM remote duster, we have selected two independent NIST dusters located on two different sites, separated by approximately 500 meters. They are connected via both Ethernet and ATM networks as shown in Figure 1.

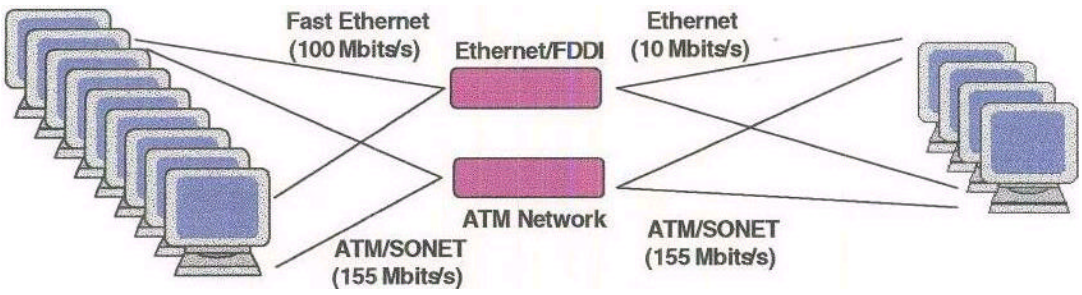

Figure 1: Configuration of the MIST remote cluster testbed

Each of the nodes on the two clusters are identical, each containing a single 200 MHz Pentium Pro Processor, with 16 Kbytes of L1 cache, 256 Kbytes of L2 cache, and 128 Mbytes of RAM. All nodes are running the Linux 2.0.29 kernel and Ohio Supercomputer Center's LAM 6.1 [BUR94] implementation of MPI (message passing interface) standard. Each machine is dual networked via two Network Interface Cards (NICs). One NIC is for 10/100 Ethernet using the Linux Ethernet device driver and the other NIC is for OC3 ATM,. Werner Almsberger in Switzerland has developed the ATM device drivers and support software as part of the Linux-ATM Application Program Interface (API) software set.

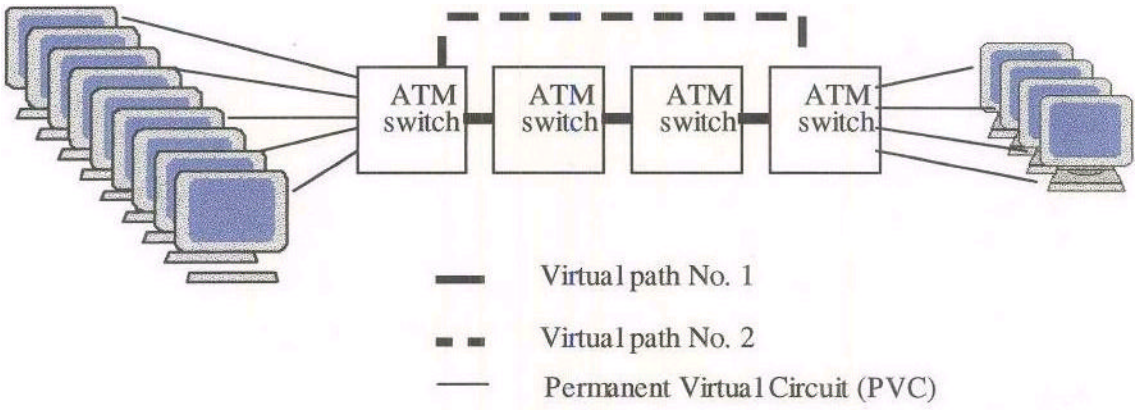

Figure 2 : ATM over SONET (155 Mbits/s)

ATM uses SONET. The UNI (User-to-Network Interface) specification defines the end user interface to an ATM network and includes the SONET, 155 Mbits/s, OC-3 transmission media:. The two clusters are connected via multi-mode fiber optic cable. As shown Figure 2, we set up a virtual path between the two exterior switches. Then we added two interior switches in order to measure the delay added by extra switches. The ATM cluster machines are configured to use Permanent Virtual Circuits (PVCs) rather than Switched Virtual Circuits (SVCs) to avoid the latency associated with virtual circuit setup, as well as problems we encountered with UNI 3.0 and UNI 3.1 signaling.

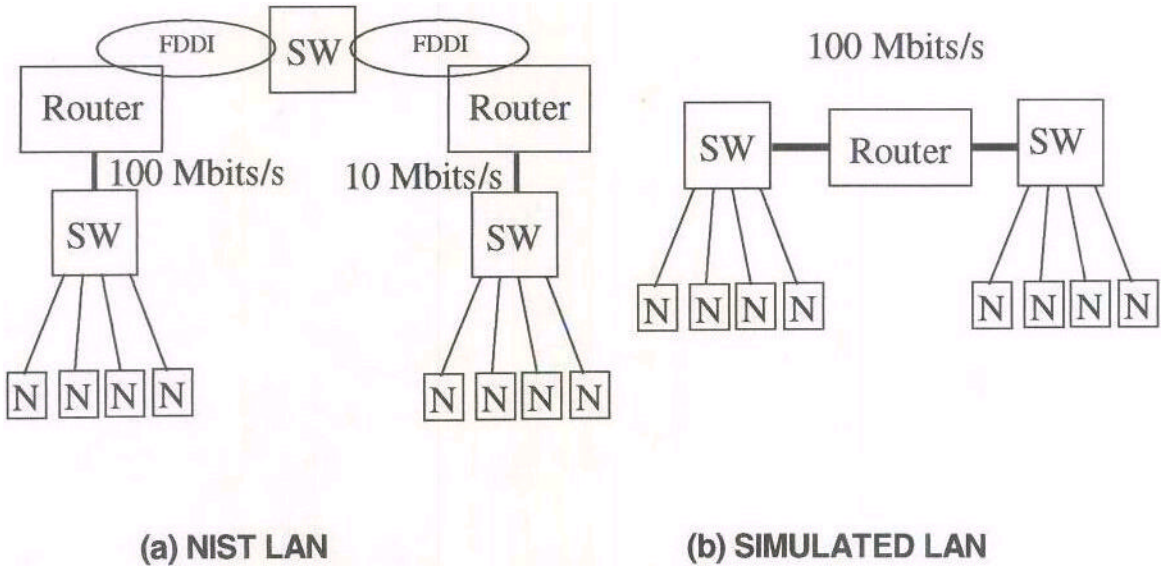

Figure 3: NIST LAN and simulated LAN Topologies.

To interface to the NIST LAN, on one side, we used SMC Ether-Express 10/100 cards and the nodes are connected to a Mega Switch 5000 HD; on the other side, we have some Intel Ether-Express Pro 10/100. The NIST LAN, see Figure 3(a), between our two remote clusters has a 10 Mbits/s link. This is a potential bottleneck and also presents an unfair direct comparison with an OC3 network. Therefore, we configured a 100 Mbits/s Ethernet LAN, see Figure 3(b) between two groups of nodes that included 2 switches and a router. This LAN was used in our experiments to simulate an all 100Mbits/s NIST LAN. The switch is a 3Com Super Stack 100 Mbit/s and the link between the two switches is a Xylan router, 100 Mbits/s.

The actual NIST LAN, with a 10 Mbits/s bottleneck link, may be representative of interconnecting clusters through the internet, where such occurrences are common. Also the actual NIST LAN can be used to simulate scaling problems as the size of the cluster and the amount of traffic grows.

2 Performance measurement tools

In addition to the standard "gettimeofday()" clock, we also used the NIST developed MultiKron® [MIN94] performance measurement instrumentation. MultiKron® is an application specific integrated circuit designed for performance data collection with low perturbation and high precision clocks. An associated toolkit [MINK97] has been developed consisting of a printed circuit board (PCB), support software, and analysis software. The PCB contains a MultiKron® chip with 16 Mbytes of dedicated memory for measurement data, some logic to control the PCB activities, and an interface to a common I/O Bus. PCB versions exist for PCI, SBus and VME bus computers. The analysis software is a series of libraries and basic data capture/reduction tools used to access and condense the MultiKron captured performance data. We have installed a MultiKron® toolkit PCB in each machine of our clusters. All of the fine time measurements were obtained by using the MultiKron.

Distributed processing poses an additional issue of correlating events on different processors. Using the Network Time Protocol [MIL92], or its variants [LEV95], results in a time synchronization of approximately 1 ms, averaged over a three hours period. This may be sufficient for coarse grain events,
but it is not for fine grain events such as communication latency and execution times. MultiKron instrumentation along with custom designed time synchronization hardware will "train" a local oscillator based on GPS pulses, to achieve a 1 !!US level of synchronization, worldwide.

The picture shown in Figure 4 represents two global time synchronizer/generator boards receiving the pulse signals from the GPS satellites and each feeding a CLOCK and a RESET signal to separate MultiKron boards

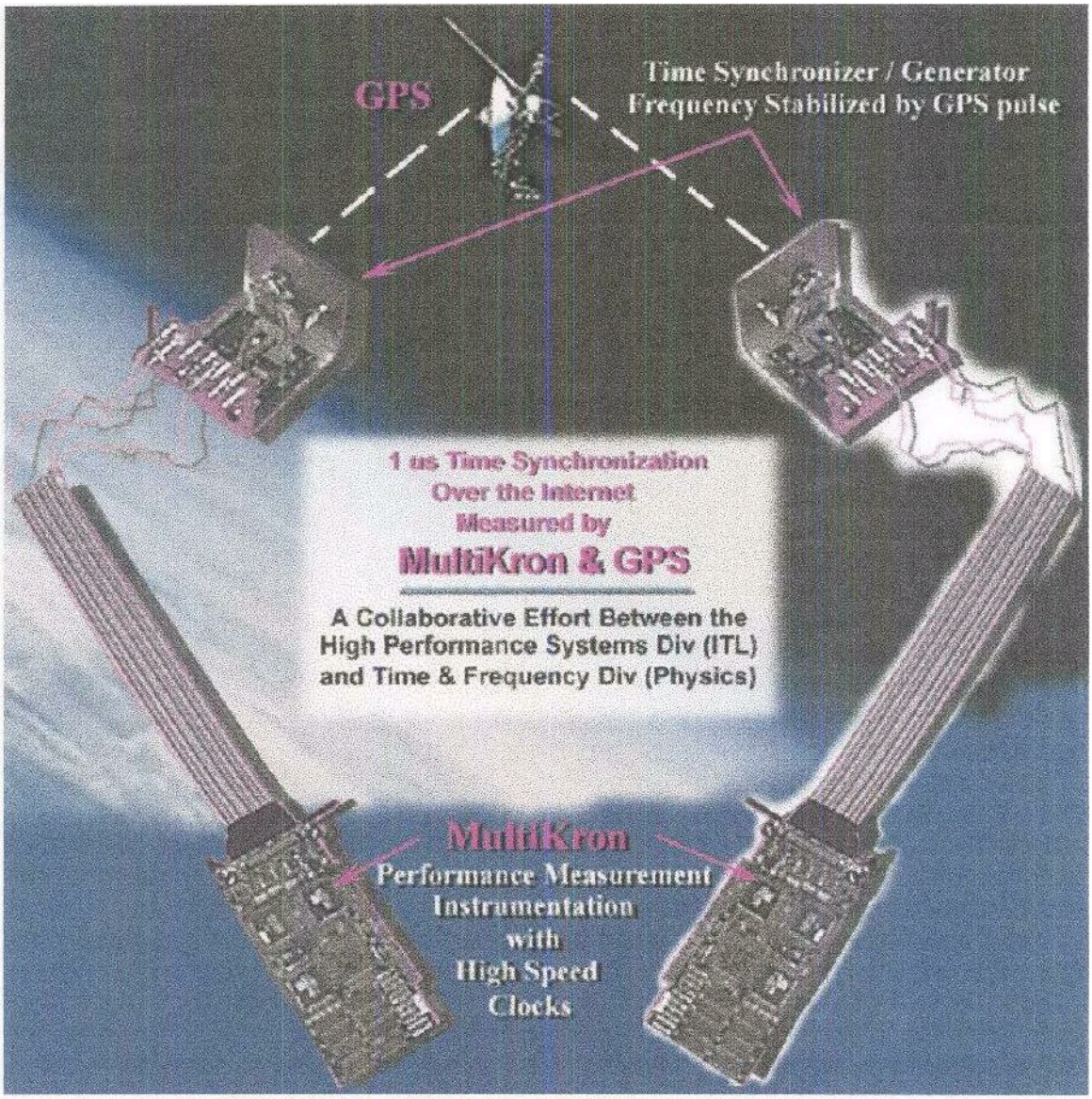

Figure 4. MultiKron Integrated with GPS based Time Synchronization instrumentation

The MuttiKron tracing facility is a low perturbation, hardware assist to the common programmer measurement paradigm. The programmer inserts (probe) code in the code to be measured. MultiKron reduces the perturbation of this instrumentation to a simple assignment statement. The user data written to the MultiKron is used to identify the location in the code, and possibly some qualifying information. MuttiKron appends to this user data a node and process ID as well as a timestamp whose resolution is 100ns. All this data is then sent off chip for on-the-fly analysis or storage and later analysis. The perturbation to the executing process is reduced to a single assignment statement.

Each performance counter of the MultiKron can be used as a stopwatch, to time software execution - with a programmable resolution down to 20 ns. They also can be used to count the occurrences of hardware signals (e.g., cache hits) or to time the duration of a hardware signal (e.g., memory bus utilization).

3 ATM Configuration

ATM is a point-to-point, switched technology. To connect two machines together-via ATM a virtual circuit has to be established between the two hosts. There are two types of virtual circuits: Switched Virtual Circuits (SVCs) and Permanent Virtual Circuits (PVCs).

Switched Virtual Circuits are connections that are established dynamically, and are then torn down when the connection is no longer needed. However, there is a high connection delay associated with SVCs due to the overhead of establishing the connection. In addition, SVCs are deleted after a timeout period if no traffic is sent over the connection. Therefore, the delay associated with SVCs is not always predictable. Also, we encountered several problems when using SVCs, such as connections not being established, or sometimes failing to remain open.

One problem we encountered with SVCs was due to the UNI signaling to the switch. It appears that the connection between two hosts will drop without either host closing the connection. The ATM signaling daemon then goes into a retry loop, attempting to reestablish the connection. A problem arises when the signaling times out with an error message. Eventually the host will lock up due to a memory leak somewhere in the execution path of signaling.

Permanent Virtual Circuits are set up once and kept open until explicitly closed. Thus, there is no latency associated with establishing the connection, as there is when using SVCs. The disadvantage of PVCs is that the switch must be preconfigured with all the connections between the hosts. When you have several hosts, and each host needs to communicate with all the others, the number of PVCs needed within the ATM switch grows quickly. In addition, PVCs are connection-oriented and unidirectional, so to allow two hosts to communicate together requires two connections.

To avoid the problems of using SVCs between nodes, and to avoid the delay due to the connection set-up time, we choose to use PVCs.

A virtual circuit is specified by concatenating 2 fields of the five-byte ATM cell header, the virtual path identifier (VPI) and the virtual channel identifier (VCI). The VPI and VCI are used to route the cell through the ATM network, see Figure 5. A cell arriving at an ATM switch at an input port with a VPI X, VCI Y will be transferred to the output port with a VPI A, VCI B based on the mapping table of the switch.

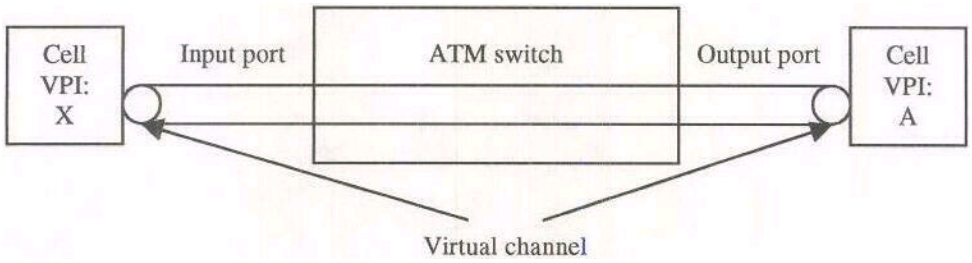

Figure 5. Composition of a Virtual Channel

To establish an end-to end PVC requires configuring pairs of entry and exit ports on each switch, and on each node configure a table of the PVCs needed to reach every other node.

The challenge was to find a simple way of establishing End-to-End PVC circuits between the nodes in the different clusters and the switches that connect them. Indeed, the problem is the number of connections we need to set. To connect a cluster of x machines to another cluster of y machines, the number of connections is 2xy. We investigated different approaches to accomplish the required configuration which consisted of using the SNMP agent of the switch, using the web interface of the switch or using the direct interface of the switch.

The switch software provides switch and connection management, IP connectivity, and SNMP network management. The Switch Control Software (SCS) controls the switch boards and handles connection set-up and tear-down duties.

We can login and open an ATM Management Interface (AMI) session, using the web interface but we do not have any way to automate the dialog configuration. Compared to a login session, the web interface presents some graphics and shows a drawing of the connections. We can set up and delete PVCs, but again, we have to enter them manually one by one.

Another approach is use the SNMP agent of the switch. The SNMP agent enables the remote monitoring and configuration of these switches. The MIB (Management Information Base) is the set of parameters an SNMP manager can query or set in the switch via an SNMP agent. The MIB is specific to this switch and we must buy the specific associated SNMP manager for this switch. The cost of an SNMP manager was not justified by our intended use as well as potential interoperability problem between different equipment.

AMI is the user interface to SCS. AMI lets users monitor and change various operating configurations of the FORE Systems switches, which include IP connectivity and SNMP network management. We decided to automate the connections using the software tool Expect. This solution appeared to be the quickest to implement and the cheapest. The concept of Expect is to write a shell script instead of typing directly the commands on the AMI of the switch. Expect can be used to automate the interactions to and from a process. It is based on three key words: SEND (sends strings to a process), EXPECT (waits for strings from a process) and SPAWN (starts a process). The language used is TCL

First, we build a table listing the correspondence between the input port, input VPI, input VCI, output port, output VPI and output VCI of the PVCs we want to set up. Then we need to run two scripts. The first script configures the endpoint switches. It logs into the switch and loops through the table setting the PVCs. We can reuse the same principle to delete the PVCs. Using the same table, the second script, which connects to each node, will create the ATM ARP entry on each node.

For the interior switches, those between the two end-point switches, the process is easier. Only the VPIs are needed to establish the correspondence between input and output switch ports, since in our case all our traffic enters on one port and exits on a second port. In this case, a single virtual path (VP) can establish the connection between the two ports while the various virtual channels pass through with VCIunchanged.

4 Performance comparison

To determine the viability of remote clusters, we ran a number of experiments to compare the performance of local and remote clusters, using our different networks.

We introduced a simulated LAN in order to be able to compare the remote clusters performances between the ATM network and Ethernet. The NIST LAN has at least one "slow" network segment (i.e.,

10Mbits/s Ethernet links or even slower T1 lines at 1.5 Mbits/s) between remote clusters. So we expected these areas to be performance bottlenecks for the NIST LAN.

Our first set of experiments focused on the network performance, we measured latency and throughput from the application viewpoint.

Latency is the time for a message to be sent from one node and received by another node. To measure the latency of message transmission, we sampled the GPS synchronized MultiKron Clocks on the send node, just prior to sending, and on the receive node, just after reception. We conducted measurements for message sizes between 4 bytes and 6 Kbytes, over various configurations of our ATM network, NIST LAN, and simulated LAN. See Figure 6.

We measured the latency with two sets of ATM NICs and their associated device drivers. We found a significant difference between the performance of these NICs in terms of latency, see table 1. The latency added by our ATM switch is approximately 18 us, and it appears to be independent of the message size. However, the latency added by the Ethernet switch indicates that each Ethernet packet (maximum 1460 bytes) is buffered before processing, since the packet of each message are pipelined in the switch, this delay only effects the first packet of each burst or message. This explains the slope change in our Ethernet curves of Figure 6 occurring at message size of 1460 bytes. The GPS synchronized MultiKron allowed us to obtain very accurate measurements.

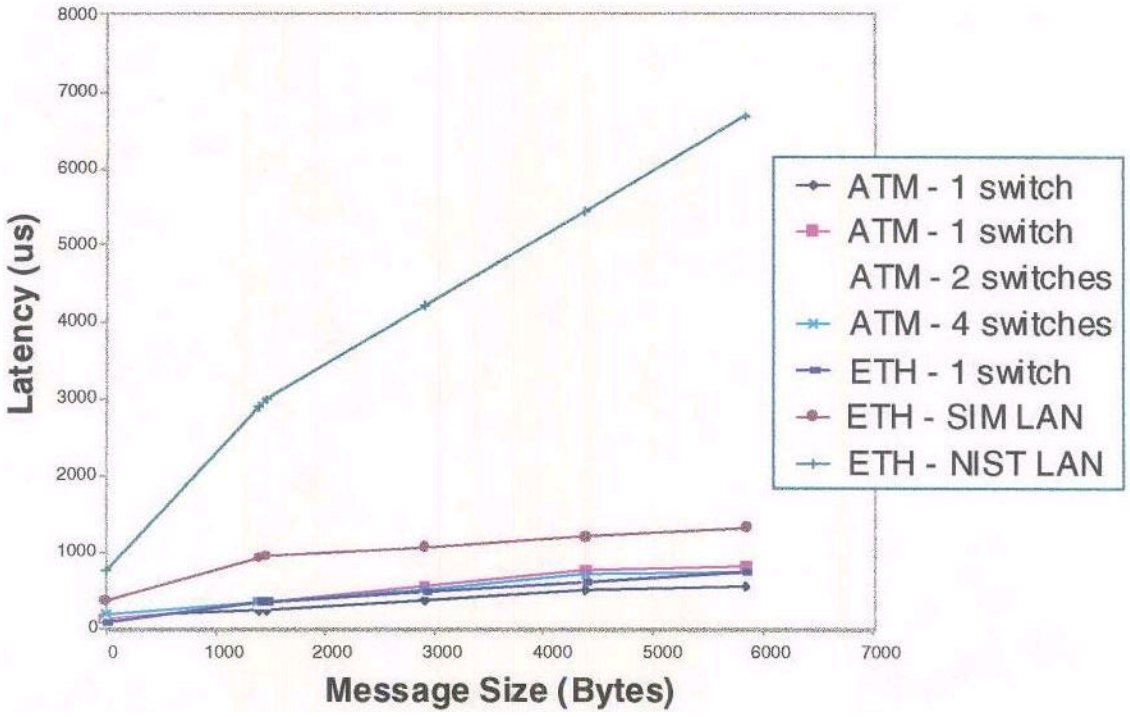

Figure 6. Message Size vs End-to-End Message Latency

SendBytes	ATM-1	ATM-1	ATM-2	ATM-4	ETH-1	ETH-SIM	ETH-NIST
	nic1	nic2					
4	139	131	164	195	82	384	757
1400	239	326	304	339	341	917	2891
1460	242	331	307	344	352	941	2982
2920	359	529	461	498	476	1050	4206
4380	494	747	645	686	596	1190	5424
5840	542	806	695	731	719	1312	6680

Table 1. Switch/ NIC Latency (us) for Different Message Sizes and Different Configurations

We next obtained throughput measurements on these various network configurations, see Figure 7. Throughput is defined as the maximum number of bytes transferred per unit time.

For ATM, the number of switches does not have much influence on the throughput, the curves are almost the same. We also notice that the maximum is reached very quickly. The throughput of the Ethernet NIST LAN is limited by a 10 Mbits/s link. The Ethernet simulated LAN throughput is lower than that for the local Ethernet network, due to the presence of the router the network cannot sustain 100 Mbits/s of traffic.

Our next set of experiments collected performance data on well known benchmark codes and some NIST applications. We used the Numerical Aerodynamics Simulation Parallel Benchmarks (NAS NPB 2.3b2). They are a set of eight benchmark problems that are designed to measure the performance of parallel computer systems for a subset of algorithms that characterize various computationally intensive aerophysics applications [BAI95]. The eight problems consist of five kernels and three simulated Computational Fluid Dynamics (CFD) applications. The benchmarks are based on Fortran 77 and the MPI standard. This suite of test programs is useful for our study since it includes programs with varying computational granularity and different message sizes.

In this section, we describe the performance results for each benchmark. We ran them on three different networks: ATM, NIST LAN and Ethernet Simulated LAN. For the benchmarks which use 8 processors, we experimented with three configurations: 8 local nodes, 6 local and 2 remote nodes, 4 local and 4 remote nodes. The different charts show the processing time and the communication time in seconds. For Figures 8-14, each bar represents the total execution time of the process. That time is further divided into processing time, as the bottom portion of the bar, and communication time, as the top portion of the bar. On the x-axis L denotes local nodes and R denotes remote nodes.

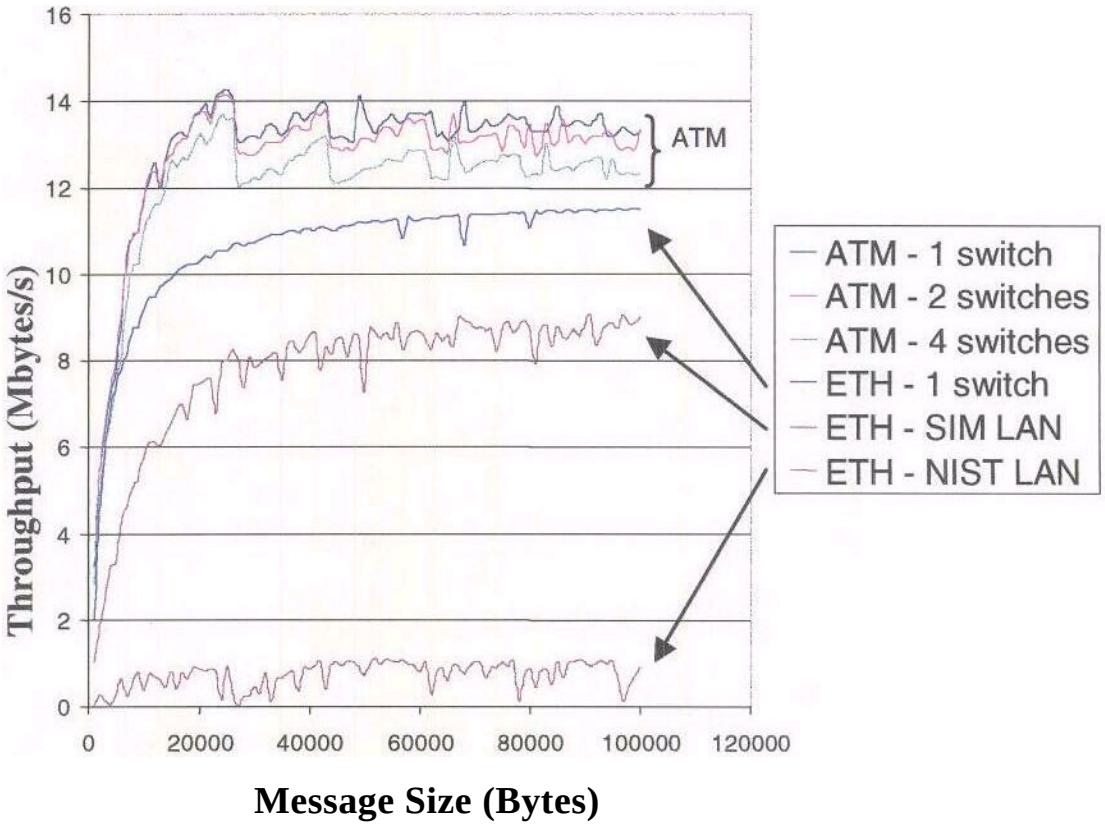

Figure 7. Throughput (MBytes/S) for Different Network Configurations.

The Multigrid (MG) Benchmark is a simplified multigrid kernel, which solves a 3D Poisson Partial Differential Equation. The problem is simplified in that it has constant rather than variable coefficients, as in a more realistic application. This code, see Figure 8, is a good test of both short and long distance communication, although the communication patterns are highly structured. For the Muttigrid benchmark each processor sends messages ranging in size from 100 bytes to 32 Kbytes.

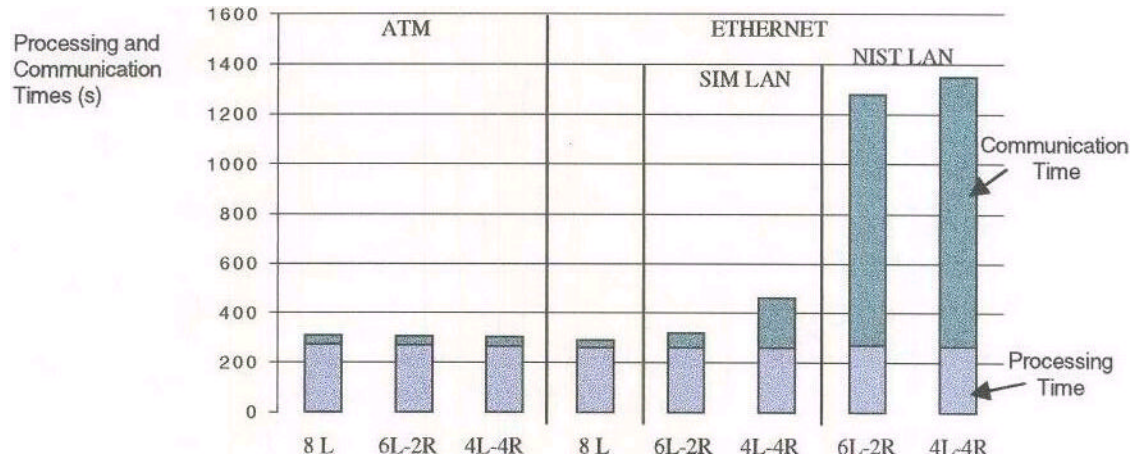

Figure 8. Remote Cluster Performance of the MG NAS Benchmark

The 3D FFT PDE (FT) Benchmark, see Figure 9, solves a 3D partial differential equation using Fast Fourier transforms. It performs the essence of many "spectral" codes, and is a good test of long-distance communication. This benchmark is unique in that it is acceptable to substitute any computational library routines. The rules of the NAS Parallel benchmarks specify that assembly-coded library routines can be used to perform matrix multiplication and 1 D, 2D, or 3D Fast Fourier Transforms (FFTs). Although FFTs send a small number of large messages (about 256 KB each), even this does not save the benchmark from being communication bound. FT is a good test of long-distance communication performance.

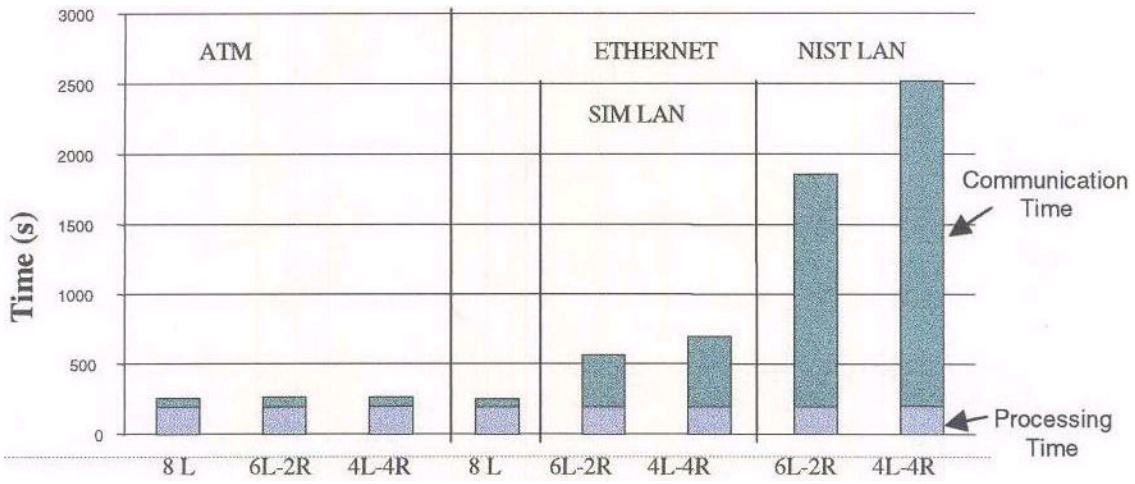

Figure 9. Remote Cluster Performance of the FT NAS Benchmark.

The Lower block-Upper block matrix decomposition (LU) benchmark, see Figure 10, does not perform an LU factorization, but instead uses a symmetric, successive over-relaxation numerical scheme to solve a regular-sparse block (5*5) lower and upper triangular system. This problem represents the computations associated with a newer class of implicit CFD algorithms, typified at NASA Ames by INS3D-LU. This problem exhibits a somewhat limited amount of parallelism compared to the other two simulated CFD applications. A complete solution of this benchmark requires 250 iterations. It sends very small messages, less than 50 bytes each.

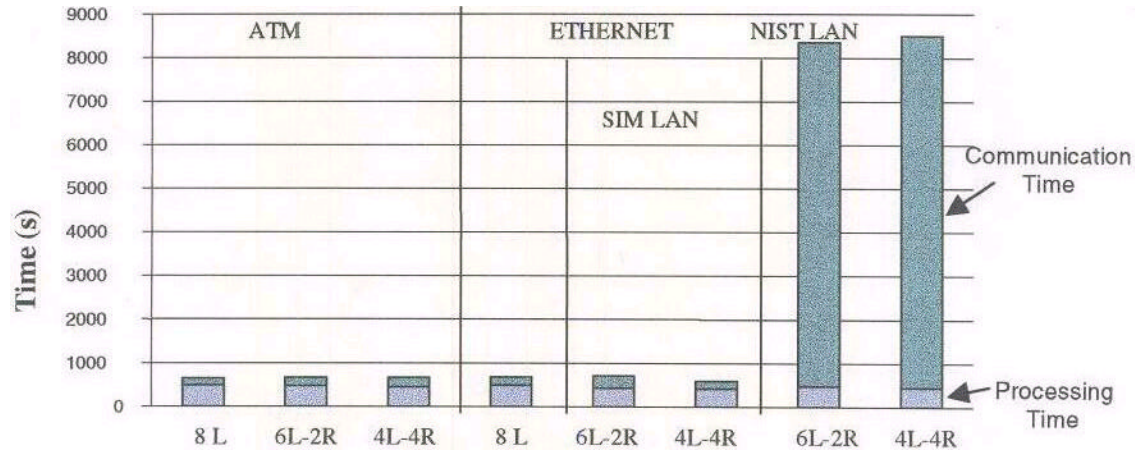

Figure 10. Remote Cluster Performance of the LU NAS Benchmark.

The Scalar Pentadiagonal (SP) benchmark, see Figure 11, solves multiple independent systems of nondiagonally dominant, scalar pentadiagonal equations. A complete solution requires 400 iterations. The Block Tridiagonal (BT) Benchmark, see Figure 12, solves multiple independent systems of non-diagonally dominant, block tridiagonal equations with a 5*5 block size. A complete solution requires 200 iterations. The BT solver and the SP solver benchmarks are both communication bound and send many medium-sized messages ranging from a few hundred to a few thousand bytes each.

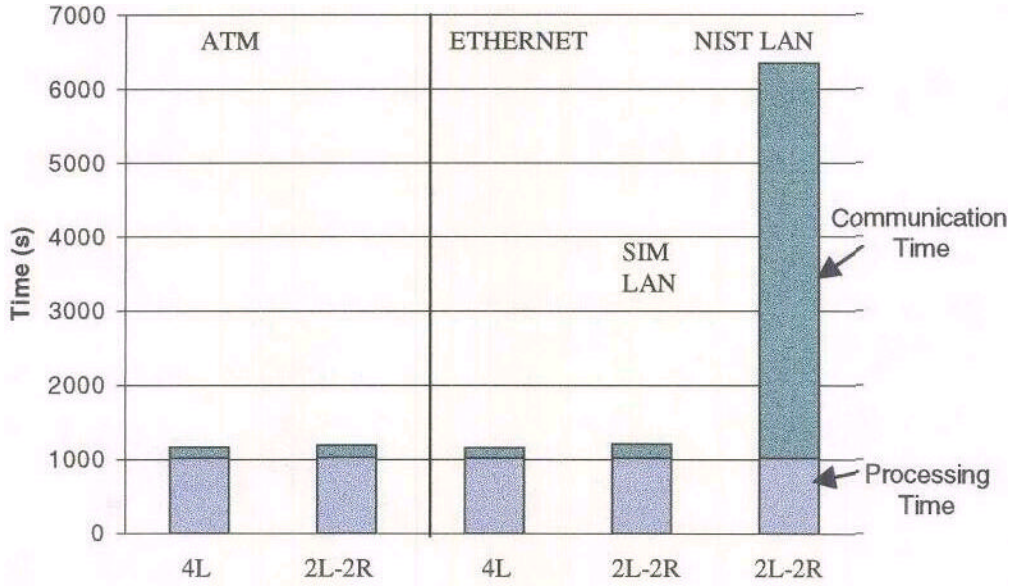

Figure 11. Remote Cluster Performance of the SP NAS Benchmark.

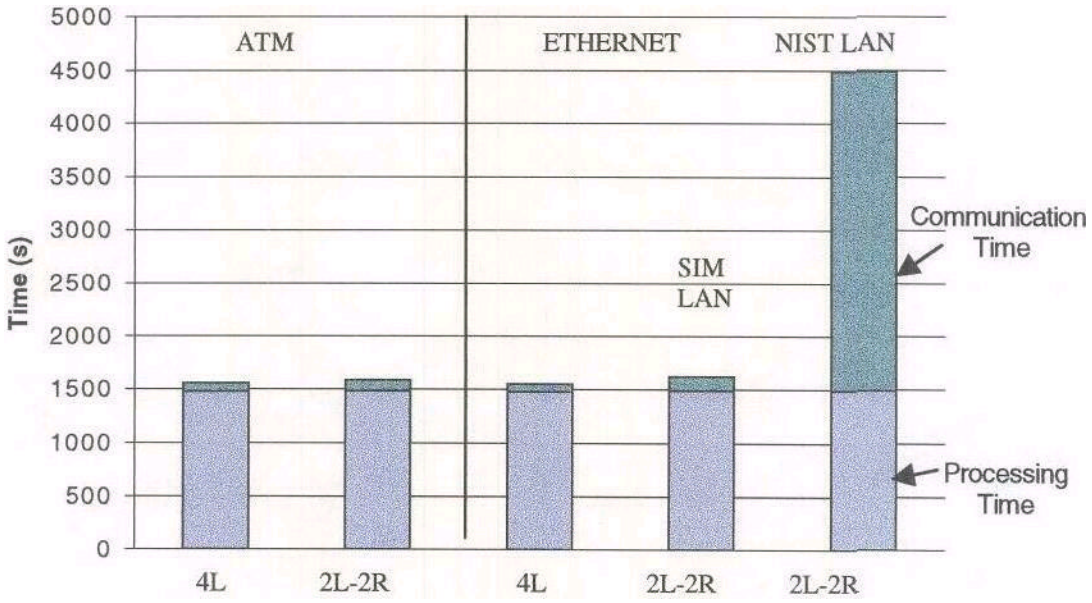

Figure 12. Remote Cluster Performance of the BT NAS Benchmark.

For all of the NAS benchmarks, using remote cluster over an ATM network does not significantly change the performances. The ATM network is stable and once the network connectivity is established, the use of remote cluster is transparent for the user. We have as our baseline the results of local cluster runs which use only fully connected local nodes via an ATM or Fast Ethernet switch. From this we see that the computation time of all our benchmarks are identical and independent of local or remote configuration. Because of the low overhead of the ATM switches, we see only small communication delays which slightly increase the overall performance of local vs remote ATM clusters.

When we compare local and remote performance for the NIST LAN, performance plummets for small messages while for larger messages the communication performance degrades more linearly. For the benchmarks which send many small messages like LU, BT, SP we see that performance is as much as an order of magnitude slower. Performance over the Ethernet Simulated LAN is comparable to using the ATM network. For this configuration moderate to large message size tends to have a negative impact on performance rather than small messages.

We next compare performance of two parallel NIST applications. All applications are written in Fortran and use MPI.

The first application is a Three-Dimensional Helmholtz Solver (3dmd), see Figure 13, which implements a matrix decomposition algorithm to solve elliptic partial differential equations, specifically the Helmholtz equation. Prior experience has determined this application to be "course grained" since it generates infrequent large messages (100 Kbytes or more). Performance advantage is observed for ATM versus Ethernet NIST LAN. This result is consistent with the results we obtained previously for large messages as FT.

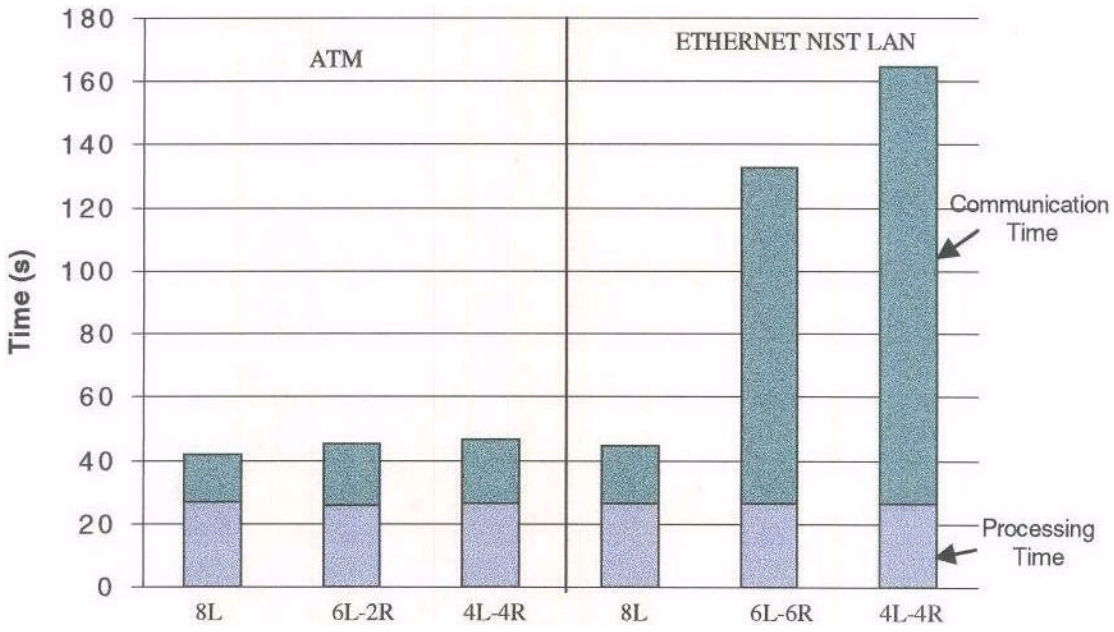

Figure 13. Remote Cluster Performance of the NIST 3dmd Application.

The second application, OA, predicts the Optical Absorption spectra of a variety of solids by considering the interaction of excitons, see Figure 14. The bulk of the computation is concerned with calculating matrix elements using an FFT Convolution method to calculate quantum mechanical integrals. Both of these applications show significant performance degradation in the remote cluster environment when using the NIST LAN, but no significant difference when using the ATM switches.

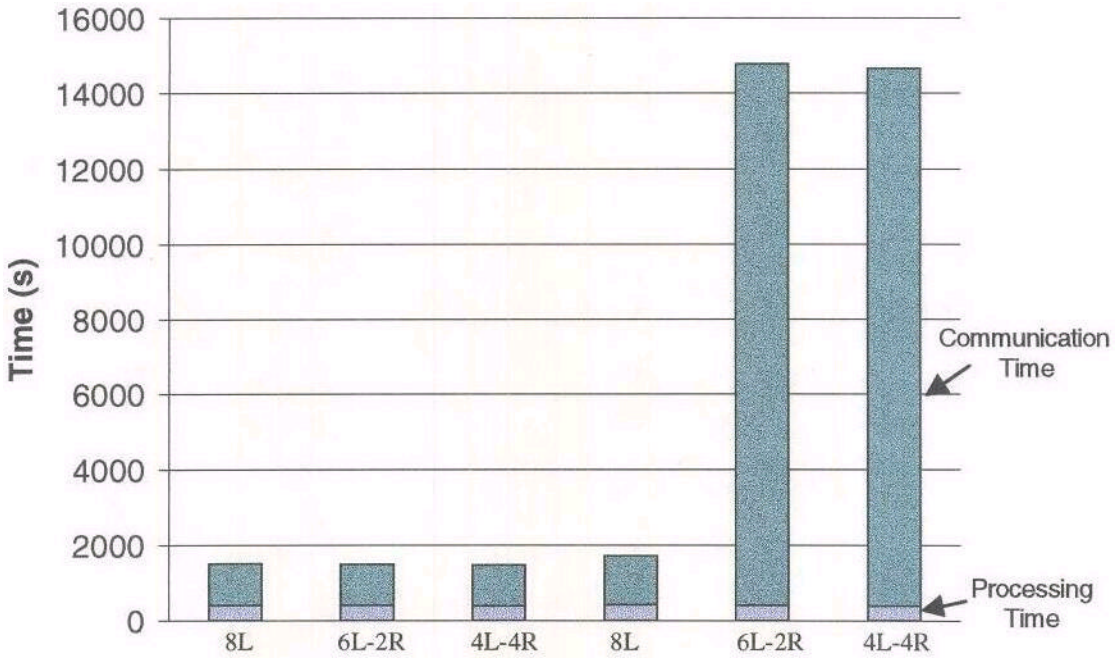

Figure 14. Remote Cluster Performance of the NIST OA Application.

Conclusion

We investigated remote cluster computing using both the Ethernet and ATM networks. We found that advanced preparation on distinct clusters significantly eases problems with different administrative domains and software versions. This is feasible since the number of different clusters is envisioned as being small.

The ATM switch configuration posed a potential problem, but we devised a methodology to configure the switches which was easily implemented through shell scripts. Performance of remote clusters is dependent on the underlying network connections. The number of hops and the types of network devices influence communication performance. Switch-based networks add little additional overhead to remote cluster computing applications. Routers add more.

We measured approximately 18 us additional latency per ATM switch (hop), resulting in essentially no degradation in throughput. Using the NIST LAN, the latency increased approximately nine times compared to the local single switch environment. Benchmark and application performance resulted in some comparable results for the local cluster using ATM or Ethernet: the local performance increase with ATM does not justify the cost of the ATM equipment. A better solution could be mixing the networks, with Fast Ethernet as the local end point switches and ATM switches connecting them, resulting in lower cost with comparable performance.

Further experiments would test larger clusters and networks, attempting to establish a rule of thumb for scalability based on the bottlenecks of a single pipe between clusters. Alternatives to a single interconnection pipe would be to use multiple pipes. This would add additional complexity in predetermining the node allocation for each pipe resulting in unbalanced communication loads where some pipes may incur delays while others are idle.

REFERENCES

[BEC95] D. Becker, T. Sterling, D. Savarse, U. Ranawake and C. Packer, *BEOWULF: A Parallel Workstation for Scientific Computation,* Proc. of the International Conf on Parallel Processing, Urbana-
Champaign, IL, Vol. I: Architecture, pp 111-114, Aug. 1995.

[LEV95] J. Levine, *An Algorithm to Synchronize the Time of a Computer to Universal Time,* IEEE Trans
on Networking, Vol 3, No. 1, pp 42-50, Feb. 1995.

[MIL92] D. Mills, *Network time protocol (version 3): specification, implementation and analysis,* DARPA
Network Working Group Rpt RFC-1305, Univ. of Delaware, 1992.

[Ml N94] A. Mink, *Operating Principles of the MultiKron II Performance Instrumentation for MIMD Computers,* NISTIR 5571, National Institute of Standards and Technology, Dec. 1994. (Available at http://cmr.ncsl.nist.gov/group.html)

[MIN97] A. Mink and W Salamon, *Operating Principles of the PCI Bus MultiKron Interface Board,* NISTIR 5993, National Institute of Standards and Technology, Mar. 1997. (Available at http://cmr.ncsl.nist.gov/group.html)

[MIN98] A. Mink, W. Salamon, J. Hollingsworth and R. Arunachalam, *Performance Measurement Using Low Perturbation and High Precision Hardware Assists,* Proc. of the 1998 IEEE Real-Time System Symposium, Madrid, Spain, pp379-388, Dec 1998. (Available at http://cmr.ncsl.nist.gov/group.html)

[SAL98] W. Salamon, *Configuring ATM Networks,* Linux Journal, No. 58, Feb 1999. (Available at http://cmr. ncsl. nist.gov/group. html)

[BAI95] D. Bailey et al, 'The NAS Parallel Benchmarks 2.0", Report NAS-95-020, NASA Ames Research Center, Moffet Field, CA Dee 1995.

[BUR94] G. Burns, et al, "LAM: An Open Cluster Environment for MPI", Supercomputing Symposium
'94, Toronto, Canada, Jun 1994.

[IND] M. Indovina, A. Mink, R. Snelick and W. Salamon, *Performance Measurement of ATM and Ethernet Computing Clusters,* Proc. of ATM Developments'98. http://cmr/scalable/publications.html

[GRIM97] A. Grimshaw, W. Wulf, et al, *The legion Vision of a Worldwide Virtual Computer,* Communication of the ACM, pp39-45, Jan 1997.

[FOX97] G. Fox and W. Furmanski, "Petaops and Exaops: Supercomputing on the Web", IEEE Internet Computing, Vol 1, No 2, pp 38-46,1997.